発表概要

JNI プログラム中のバグ発見

近藤　豪[†1]　小野寺　民也[†1]

この発表で我々は，Java Native Interface（JNI）を用いたプログラムからバグを発見するための静的解析技術を述べる．JNI プログラミングは面倒でエラーが発生しやすい．なぜならコンパイラで検出できない JNI 特有の間違いが多数あるからである．この発表ではその中から次の 3 つに着目する．まず，ネイティブコードでは，例外を扱うための明示的な文が Java メソッド起動文の後に挿入されなければならない．しかし，この文は忘れられることがある．我々はこの欠陥を発見するためのタイプステート解析を提案する．第 2 に，ネイティブコードは Java VM 内にリソースを割り付けることができるが，それらは Java と違い明示的に解放されなければならない．このリソース管理におけるエラーはリークなどを引き起こす．この問題に対処するために我々は一般のメモリーエラー検出にも使われているタイプステート解析を用いる．第 3 に，Java リソースへの参照が複数回のネイティブメソッド呼び出しにまたがるときには，それは大域参照へ変換されなければならない．しかし，プログラマはこのルールを忘れて，後で使用するために局所参照を大域変数へ代入してしまうことがある．我々は，このふさわしくない慣習を検出するための構文チェッカを提供する．そして我々は，これらの解析技術を BEAM と呼ばれるバグ発見器の上に実装して，JNI を含むオープンソースソフトウェアに対して実行してみた．実験では 84 個の JNI 特有のバグをオーバヘッドなしに発見し，合わせて 73%バグ発見数を増やすことができた．

Finding Bugs in JNI Programs

Goh Kondoh[†1] and Tamiya Onodera[†1]

In this presentation, we describe static analysis techniques for finding bugs in programs using the Java Native Interface (JNI). The JNI is both tedious and error-prone because there are many JNI-specific mistakes that are not caught by a native compiler. This presentation is focused on three kinds of common mistakes. First, explicit statements to handle a possible exception need to be inserted after a statement calling a Java method. However, such statements tend to be forgotten. We present a typestate analysis to detect this exception handling mistake. Second, while the native code can allocate resources in a Java VM, those resources must be manually released, unlike Java. Mistakes in this resource management cause leaks and other errors. To detect Java resource errors, we used the typestate analysis also used for detecting general memory errors. Third, if a reference to a Java resource lives across multiple native method invocations, it should be converted into a global reference. However, programmers sometimes forget this rule and, for example, store a local reference in a global variable for later uses. We provide a syntax checker that detects this bad coding practice. We have implemented our analysis techniques in a bug-finding tool called BEAM, and executed it on opensource software including JNI code. In the experiment, our analysis techniques could find 84 JNI-specific bugs without any overhead and increased the total number of bug reports by 73%.

（平成 20 年 5 月 22 日発表）

†1 日本アイ・ビー・エム東京基礎研究所
Tokyo Research Laboratory, IBM Research

© 2008 Information Processing Society of Japan